Brave Mobots Use Representation

Chris Thornton
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QN
UK
Email:

This paper launches an all-out assault on this interfacing problem. It introduces a novel learning method ('explicitation') and shows how its unsupervised, constructive nature allows it to be used to link learning with both evolutionary and

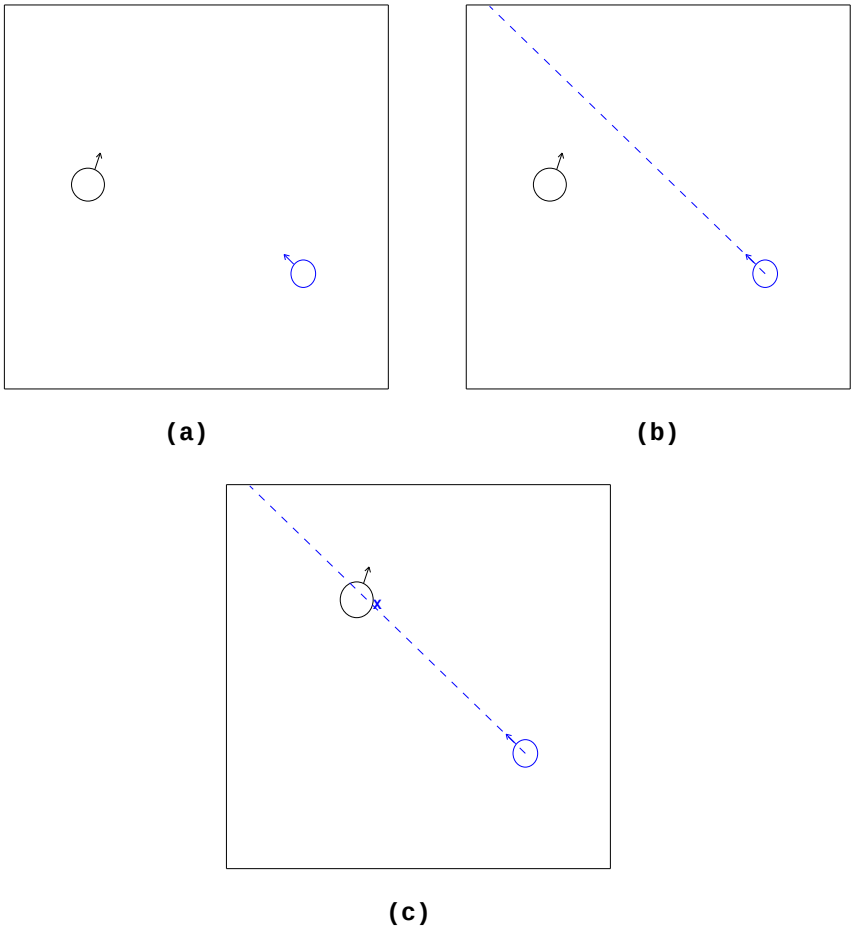
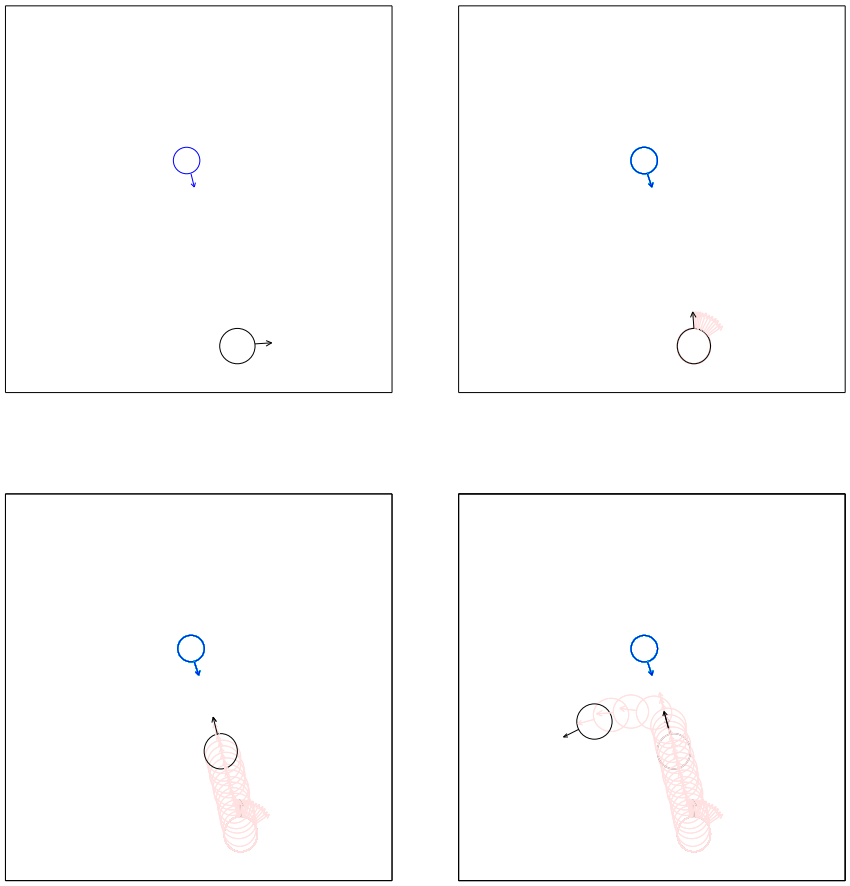


Figure 2: Basic agent scenarios.

previous sensor input). In each time cycle of the simulation, the sensor produces a real value between 0 and 1 which varies monotonically with the *proximity* of the nearest obstacle along a ray pointing directly ahead, see Figure 2 (b). The border of the arena does not constitute an obstacle and thus never affects the sensed proximity value. However, it may reflect the proximity of the other agent if the position of that agent falls somewhere along the sensor ray, see Figure 2 (c). If no obstacle intersects the sensor ray, the sensor returns a zero value. The larger of the two agents, known as the **predator**, has a variety of sensors. However, since the model is concerned with the development of behaviour in the prey, these are not relevant and will not be discussed.

Within the simulation, the predator's goal is to to destroy the prey as quickly as possible. The prey's goal is to survive while, at the same time, minimising movement. The predator attempts to achieve its goal by implementing a 'search-and-destroy' strategy. If it finds itself facing away from the prey, it turns towards it; see Figure 3 (a) and (b).



a)

itself (cf. Bolles, 1979; Flaherty, 1985). Where the issue of the development of behavioural responses is confronted explicitly (cf. Barnett, 1973), there are few operational models and none as yet that deal explicitly with fight-or-flight.

Generic work on the evolution of intelligent behaviour has typically focussed on the genetic algorithm model of Holland (Holland, 1975; Goldberg, 1989) and the classifier system of Wilson (1991). Work on behaviour learning has concentrated on algorithmic models such as recursive decision-tree generation (Quinlan, 1986) or neural-network models such as backpropagation (Rumelhart, Hinton and Williams, 1986) or competitive learning (Rumelhart and Zipser, 1986). However, there has been no previous attempt to produce a computational model which shows the role played by evolutionary, learning and representation-construction processes in the development of this behaviour.

4 Evaluating standard learners

Can the development of the fight-or-flight behaviour in agents be modelled *solely* in terms of lifetime learning? To investigate this various learning methods were tested for their ability to acquire the fight-or-flight response. Experiments were conducted using C4.5[1] and standard backpropagation. Both of these are supervised methods and thus require explicit training examples. In the experiments carried out a training set of 1000 input/output pairs was used and these were derived direct from a running simulation.

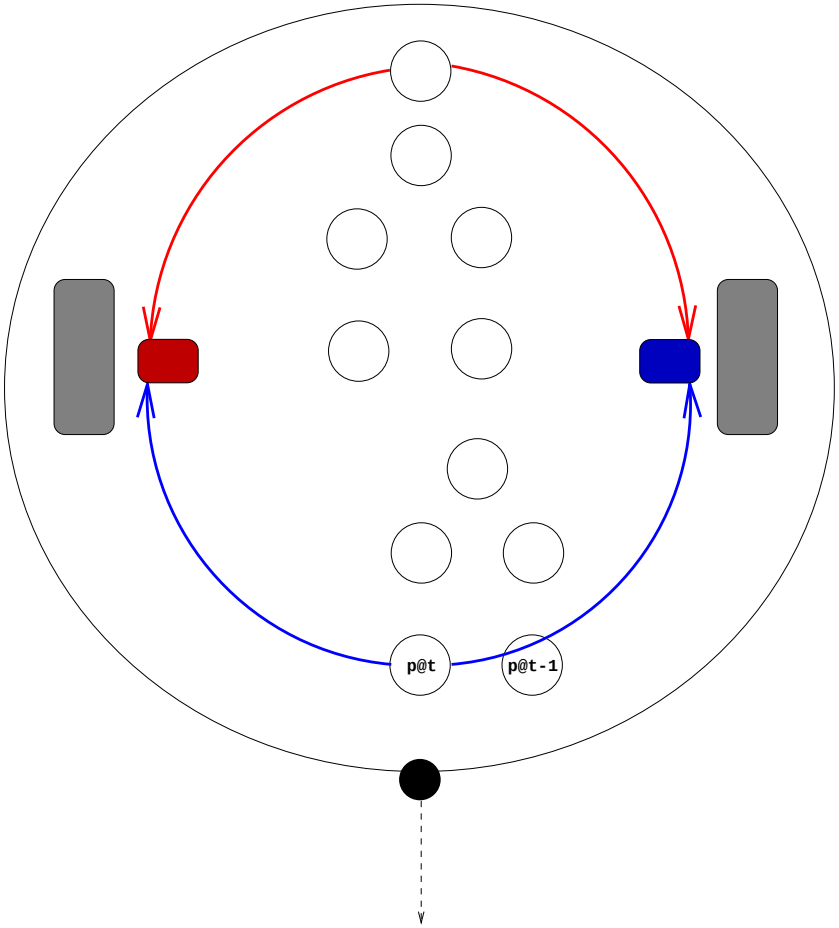
The input in each pair was a vector of numbers corresponding to the prey's current sensory input and its corresponding input in the previous cycle. (The second input constitutes the learner's 'memory' of the first.) The output in each case specified the wheel motions for an appropriate execution of the fight-or-flight behaviour. Thus, in those cases where the input vector was derived in the context of an oncoming, accelerating predator, the target outputs specified the wheel motions for a flight response. In the testing of backpropagation standard parameters for learning rate and momentum (0.01 for learning rate and 0.9 for momentum) were used in an ordinary feed-forward architecture of three, fully interconnected layers. Architectures involving from 3 to 30 hidden units in the middle layer were tested, but it was found that the number of hidden units made little difference to the final performance.

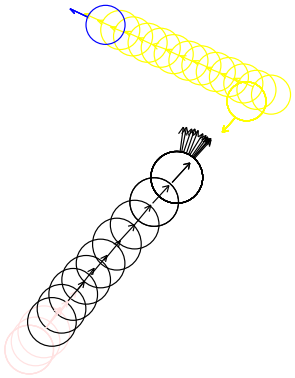
The two methods were both tested for their ability to generalise. The quality of generalisation with respect to unseen cases was examined, as was the degree to which the learning methods were able to reproduce the fight-or-flight response in a replication of the original simulation. In both cases, both methods performed poorly. The average results obtained are summarised in Table 1. The error shown here is RMS error on a testing set of 1000 cases. The 'Deaths' column shows the number of deaths sustained by a trained agent in a simulation of 2000 time steps, in which the predator agent produced 12 aggressive (accelerating) approaches.

	Error	Deaths
C4.5	0.209	12
Backpropagation	0.670	8

Table 1: Performance of C4.5 and backpropagation on fight-or-flight.

[1] C4.5 is an improved version of the well-known ID3 algorithm (Quinlan, 1986).


p@t
p@t-1



$$P(x_i = v | g(X) = v_g)$$

Here X is the entire datum and v_g is the value of a function g, which evaluates the implicit property.

Methods which attempt to discover and exploit such probabilities for inductive purposes, without using any other source of information, are 'empirical learning' algorithms. There are a large number of these (Shavlik and Dietterich, 1990, Michalski, Carbonell and Mitchell, 1983, Michalski, Carbonell and Mitchell, 1986). However, the Bayesian analysis enables us to divide them up into two basic types.

A method that attempts to exploit either of the first two forms of probability confronts a relatively easy task. Only cases that are *explicitly* observed in the data need to be taken into account. There are a finite number of these. The task thus involves deriving frequency statistics (probabilities) over a *finite* dataset.

A method that attempts to exploit probabilities of the third form confronts a much harder task. It has to first identify the appropriate evaluation function for the implicit property (i.e., it has to guess what the property is). There are an *infinite* number of possible implicit properties and the task thus involves dealing with an infinitely large search space.

Practical learning methods naturally tend to be predisposed towards the easier task, i.e., they tend to exploit probabilities of the first and second form. A typical example is the Focussing method (Bundy, Silver and Plummer, 1985). Some methods such as ID3 (Quinlan, 1986) do not consider the third form at all. On the other hand, there are also methods which focus exclusively on the third form. Examples include the 'BACON-esque' methods of Langley and co-workers (Langley, 1977; Langley, 1978; Langley, Bradshaw and Simon, 1983; Langley, Simon, Bradshaw and Zytkow, 1987) and related methods such as (Wolff, 1978; Wolff, 1980; Lenat, 1982; Wnek and Michalski, 1992). Some methods such as backpropagation (Hinton, 1989) appear to straddle the fence, showing some ability to exploit both main forms (Thornton, 1994b).

Interestingly, we can deduce that the evaluation function used in the third form must measure a *relational* property of its inputs. To understand why, we need to think about the way in which the function differentiates different types of input. Let us say that the function produces a particular value whenever the input variables have certain *absolute* values. In this case, this evaluation is effectively a label for an explicit case. If all the values of the function are derived this way, the conditional probability can obviously be reduced to a set of probabilities of the second form. Thus, if the probability is a valid example of the third form, the evaluation function must measure a non-absolute (i.e., *relational*) property of its inputs. Learning problems whose solutions involve exploiting probabilities of the third form are thus **relational**. Problems which involve exploitation of probabilities for explicit cases are **statistical**, since they simply involve the derivation of frequency statistics over a finite dataset.[2]

Of course, since the space of relational effects is infinitely large, relational learners always and necessarily have a *bias* (Utgoff, 1986), i.e., they have a predisposition to consider certain types of relationship. Relational learners are also always potentially *recursive*. The identification of any set of relational effects involves the application of evaluations (functions) to the original data. This creates new values, and thus new data. These new data can themselves be processed for statistical and relational effects in a recursive manner.

At each stage, this process is effectively building a new level of description of the original data. Each level encodes or expresses a relational effect in the form of a single variable using an evaluation (a test

[2] Learning methods can be classified the same way.

or measure) of the underlying relationship. Ths structure that will be built in a given case depends not only on the original data source but also on the bias of the method. Moreover, the influence of the bias 'accumulates' and becomes increasingly strong as the process builds layer upon layer.

I call any which exploits relational and statistical effects in this recursive manner an **explicitation process**, on the grounds that it incrementally renders implicit properties of the data *explicit*, through a process of recursive redescription.

6.1 A hybrid im lementation

The explicitation process can be implemented in many different ways. However, for present purposes a 'hybrid' implementation (Torrance and Thornton, 1991) was used. This incorporates a neural-network component and an algorithmic or 'symbolic' component. The neural network component has the task of exploiting statistical effects and the algorithmic component has the task of implementing the recursive exploitation of relational effects. The latter uses a bias which effectively restricts its attention to relationships involving constant differences. A set of data were considered to exhibit a relational effect (i.e., to belong to a relationship) just in case they could be arranged into a linear order such that each variable would show a constant difference from datum to datum. Data satisfying this constraint were said to exhibit a **linear signature**. (As an efficiency measure, the relational exploitation was directed towards the statistical effects identified at any given layer, rather than to the relevant original data. Thus the learning always adopted a coarse-grained view of available data.)

The foundation for the neural network (statistical) component was the well-known unsupervised procedure of **competitive learning** (Rumelhart and Zipser, 1986). This is a general and robust method. However, like many unsupervised processes, it has a blind-spot in that it is unable to directly exploit low-order[3] statistical effects.[4] The basic procedure of an unsupervised learning method involves grouping inputs together according to similarity. This process 'discovers' cases in which a set of input values typically co-occur and thus tends to expose nth-order associations between variable values (where n is the number of values making up a complete input). However, the process *ignores* possible associations of order less than n, i.e., ones which do not involve the complete set of input variables taken together.

Unsupervised methods can usually be adapted to overcome this deficiency. In the case of competitive learning (Rumelhart and Zipser, 1986) the adaptation is straightforward;

where r is the learning rate.

To sensitise the method to lower-order, probabilistic structure we associate each weight with an addi-

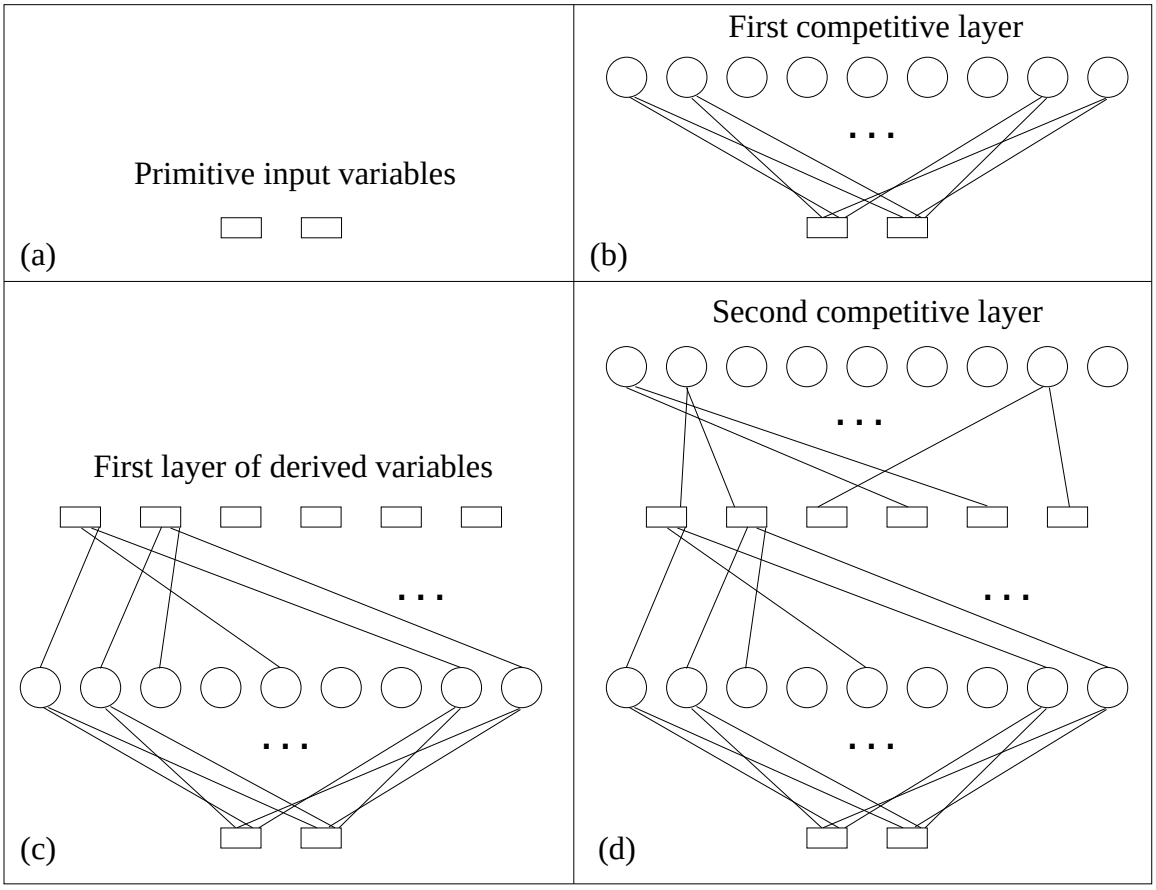


Figure 6: Explicitation using neural-network construction.

- The identification of a linear signatures at any layer always leads to the production of a new node at the next level of the network. The action of the network is configured so as to instantiate such variables with a value reflecting the geometric projection of the current input upon the line carved out by the relevant signature. Such variables thus provided an approximate measure of the relevant relationship.
- The learning is fully incremental. Nodes and layers are added to the network up until the point at which all effects — statistical and relational — are fully exploited.
- The learner's one-cycle memory is implemented by providing each main variable with a buffer which always hold its previous value.

7 The model

We now turn attention to the model itself. This takes the form of a simulation program written in the language POP-11 (Barrett, Ramsay and Sloman, 1985). The program takes about 10 minutes to run

on a single-user Sun SPARC 1+. As we will see, the phenomena it generates while doing so include simulations of evolutionary processes, learning processes and interactions between predator and prey agents.

The simulation divides up into a number of phases. In the initial phase, the world contains a single prey agent and a single predator agent. The predator agent exhibits the 'non-aggressive' behaviour pattern described above. If it finds itself facing away from the prey, it simply attempts to turn towards it; see Figure 3 (a) and (b). If the predator finds itself facing directly towards the prey, it moves forward at a 'slow' or a 'fast' pace: Figure 3 (c).

are excitatory connections from the sensor to both wheel motors. With these biases and sensor-motor connections, the agent will tend to move towards whatever excites its sensor, tending to veer to the left at all times (Braitenberg, 1984).

During the initial phase, the prey is subject to an evolutionary process involving asexual reproduction. The prey's internal architecture is specified using a simple genetic encoding, i.e., a 'genotype'. Each

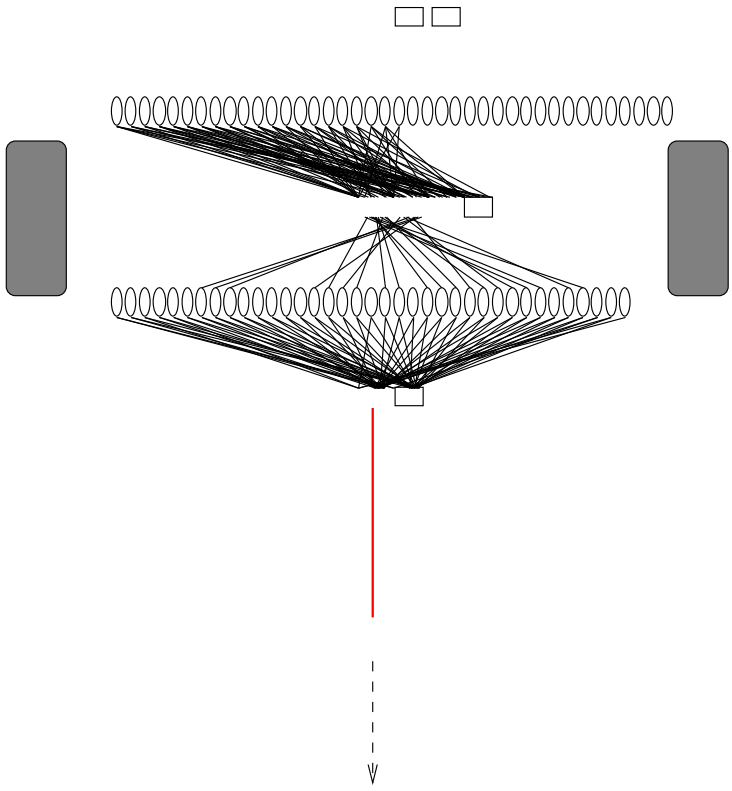
7.1 Events in the initial learning phase

As we have seen, the prey has a single sensor which senses the proximity of the nearest obstacle (e.g., the predator) along a ray pointing directly ahead. Predator attacks, in the initial phase of the model, take the form of rapid advances towards the prey, which are then aborted if the prey turns out to be facing towards the oncoming predator. Thus the prey's input environment is a sequence of proximity values, a sample of which is as follows.

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59
0.59 0.59 0.59 0.59 0.60 0.60 0.61 0.62 0.63 0.64 0.66 0.67 0.68
0.69 0.70 0.71 0.72 0.73 0.74 0.76 0.77 0.78 0.79 0.80 0.81 0.82
0.84 0.85 0.86 0.87 0.88 0.91 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

Note the prevalence of zero (0.00) values, the runs of increasing values (signifying an oncoming predator) and the runs of identical values (signifying a predator moving into a head-on position).

Learning takes place in the prey using the hybrid explicitation method. As mentioned, all data variables in the learning process are buffered, which means that the learning actually has access to a data stream based on two variables: the sensor value itself and the previous sensor value. The statistical-exploitation aspect of the learning leads to the discovery that particular pairs (i.e., two-value sequences) of sensor values occur with high frequency. Examples include (0.63, 0.64) and (0.77, 0.78). The existence of these is a consequence of the fact that the predator tends to advance on the prey at a steady rate, thus



at a particular proximity; the label used is thus `approach@c`. (Note that all nodes at this level respond to events extending across time `t` and `t 1` and thus have no time subscript.)

The variables making up the third level of the network are derived from the signatures detected in the effects captured by the second-level nodes. They thus measure the proximity of an approach occurring at a particular rate (i.e., either fast or slow). The labels applied are thus `fast approacher@t`, `slow approacher@t` etc. The nodes at the fourth layer of the network respond to approaches which maintain a constant or a changing description over the third-layer variables, i.e., which either stay slow or fast, or change from one rate to the other. I therefore use the labels

9 Final comments

The paper has presented a model of the development of a fight-or-flight behaviour in a simulated agent. Because the model makes use of the explicitation learning method, which is both unsupervised and constructive, it is able to present a developmental picture in which evolutionary processes effectively co-operate with learning processes in the formation of representational structure. Because the model uses processes which are fully incremental, it is able to show how an agent/environment interaction can modulate and guide an underlying evolutionary process. And because the learning is implemented in the form of a neural-network process, the learning process is at least reconcilable with current models of brain mechanism.

The model thus provides a 'big-picture' story about the way in which the development of complex intelligent behaviours might involve evolutionary processes, learning processes, agent/environment interaction and representation development. The learning method forms a a bridge between the GA paradigm on the one hand and the representationalist paradigm on the other.

Unfortunately, in its present manifestation the model has many shortcomings. As a computational implementation, it is not as robust as one would like. It also incorporates as ad hoc features the device of one-cycle memory (i.e., data buffering). The selection of a linear-signature bias during relational learning is also weakly motivated. But the most serious deficiency is probably the fact that the model makes use of what is in effect, a pre-scripted sequence of events. It requires that the predator implement certain changes in behaviour at certain points in the process. If these behavioural alterations do not take place on cue, the learning process is derailed.

However, it is believed that all these deficiencies will be remedied in the ongoing development of this work. Future versions of this model will aim to show how a fight-or-flight response emerges in a more natural, unscripted scenario, involving multiple predator and prey agents in addition to other forms of process and contingency. Work is currently in progress towards this end.

10 Acknowledgements

Thanks to Dave Cliff and Jim Stone for helpful input on an earlier draft.

References

[1] Barnett, S. (1973). *Ethology and Development.* Heinemann Medical.

[2] Barrett, R., Ramsay, A. and Sloman, A. (1985). *POP-11: A Practical Language for Artificial Intelligence Programming.* Chichester: Ellis Horwood.

[3] Boden, M. (1979). *Piaget.* Fontana Modern Masters, Fontana Press.

[4] Bolles, R. (1979). *Learning Theory* (2nd edition). New York: Holt.

[5] Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* London: The MIT Press.

[6] Bundy, A., Silver, B. and Plummer, D. (1985). An analytical comparison of some rule-learning programs. *Artificial Intelligence, 27,* No. 2 (pp. 137-81).

[7] Clark, A. and Karmiloff-Smith, A. (1993). The cognizer's innards: a psychological and philosophical perspective on the development of thought. *Mind and Language*, *8*.

[8] Cliff, D., Husbands, P. and Harvey, I. (1993). Evolving visually guided robots. In J. Meyer, H. Roitblat and S. Wilson (Eds.), *From Animals to Animats:*

[26] Michalski, R., Carbonell, J. and Mitchell, T. (Eds.) (1986). *Machine Learning: An Artificial Intelligence Approach: Vol II.* Los Altos: Morgan Kaufmann.

[27] Muggleton, S. (Ed.) (1992). *Inductive Logic Programming.* Academic Press.

[28] Nolfi, S., Floreano, D., Miglino, O. and Mondada, F. (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. In R.A. Brooks and P. Maes (Eds.), *Proceedings of Artificial Life IV* (pp. 190-197).

[29] Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1* (pp. 81-106).

[30] Reynolds, C. (1994). Evolution of corridor following behavior in a noisy world. In D. Cliff, P. Husbands, J. Meyer and S.M. Wilson (Eds.), *Proceedings of the Third International Conference on Simulation of Adaptive Behavior* (pp. 402-410).

[31] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature, 323*